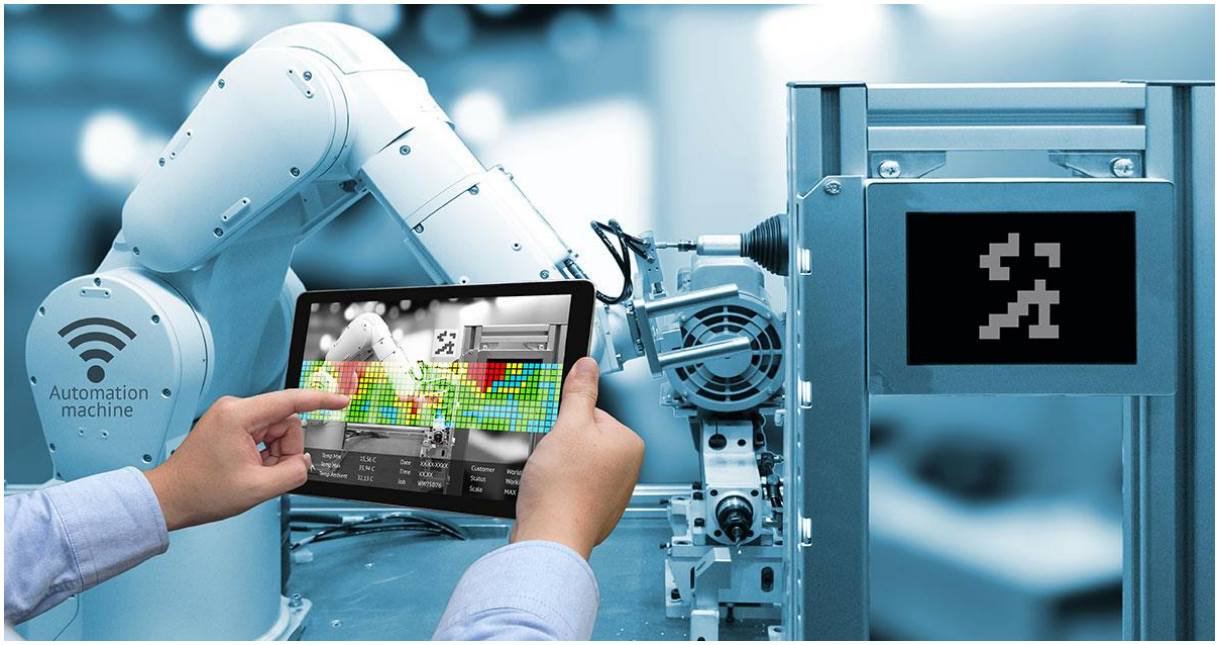

INTRODUCTION TO DDS

Factory of the Future

Version 1.1
11-01-2023

Authors:

Colin Stulp	colin.stulp@windesheim.nl
Hein Kamphuis	hein.kamphuis@windesheim.nl
Jurn Kloosterman	jurn.kloosterman@windesheim.nl
Mathis Plassart	mathis.gabriel.plassart@windesheim.nl
Timothee Perello	timothee.perello@windesheim.nl

Table of contents

Introduction

This paper is meant for other researchers and interested people who are going to investigate the Data Distribution Service (DDS) protocol. The goal of this paper is for the reader to get a quick understanding of how DDS works and how it is used in practice.

The first chapter explains how the DDS protocol works and why it is used in different infrastructures. It also explains the architecture of DDS and the architecture of a databus. In the second chapter some use cases are given. Finally, the third chapter contains different vulnerabilities in DDS and some measurements.

1. Understanding DDS

DDS is a middleware technology that drives billions of devices and mechanisms, such as railways, autonomous cars, airports, spacecraft, diagnostic imaging machines, industrial robots, and military tanks. DDS is based on the publish-subscribe paradigm, helping the development of middleware layers for machine-to-machine communication. Maintained by the Object Management Group (OMG), DDS is used in all classes of critical applications to implement a reliable communication layer between sensors, controllers, and actuators. DDS was invented for such systems, with a strong focus on interoperability and fault tolerance. DDS is optimized for publish-subscribe and peer-to-peer applications because these kinds of applications cannot afford a single point of failure. (Trend Micro Research, 2022)

Development of DDS started in 2001. It was developed by Real-Time Innovations (RTI), a software framework company, and Thales Group, a French defense company. In 2004, the Object Management Group (OMG) published DDS version 1.0. (Wikipedia)

Figure 1: Train railway that works on DDS.

Figure 2: Airports that work with DDS.

1.1. Functioning of the DDS protocol

The main goal of DDS is to share the right data at the right place at the right time, even between time-decoupled publishers and consumers. Middleware should only deliver the data that consumers really need. DDS can understand the schema of the shared data. This allows it to filter on content, age and/or lifecycle to provide applications with only the data they need. For example, you can send only boiler temperatures over 300 (content filter) with at most ten updates per second (rate). This efficient approach can often save 90% of the data communications overhead in many systems. (DDS-Foundation)

Data must be available where it is needed, to facilitate self-forming systems. DDS distributes and maintains data, so it is readily available. DDS dynamically discovers publishers and subscribers, the data types they want to share, and the related Quality-of-Service (QoS). Following a successful match, DDS enforces timely distribution according to the QoS. A DDS subscriber can be sure that its peer publisher is alive, and that any data produced will be delivered. This greatly simplifies application development and error handling. (DDS-Foundation)

Real-time systems interact with the real world. Data must be available on time – the right data too late is a failure. Data differ in priority, reliability, timing, and other non-functional

properties. DDS balances utilization of scarce resources to distribute data at the right time. (DDS-Foundation)

For example, when the artificial intelligence (AI) of a self-driving car needs to issue a “turn right” command, DDS is used to transport the command to the “brain” of the car to steer the car in the right direction. The brain of a self-driving car is the electronic control unit (ECU). The same instance happens when speed sensors send information from the motor up to the ECU. This technology can prevent a self-driving car from speeding or creating accidents. (Trend Micro Research, 2022)

1.2. DDS Terminology

To get a better understanding of DDS there are some important terms you need to know. The terms are described below, to make it clearer all the terms are applied to a situation. In this example DDS is the factory, in this factory there are two floors, each floor contains four rooms, and each room contains 2 sensors with one cobot. (Helmer, 2021)

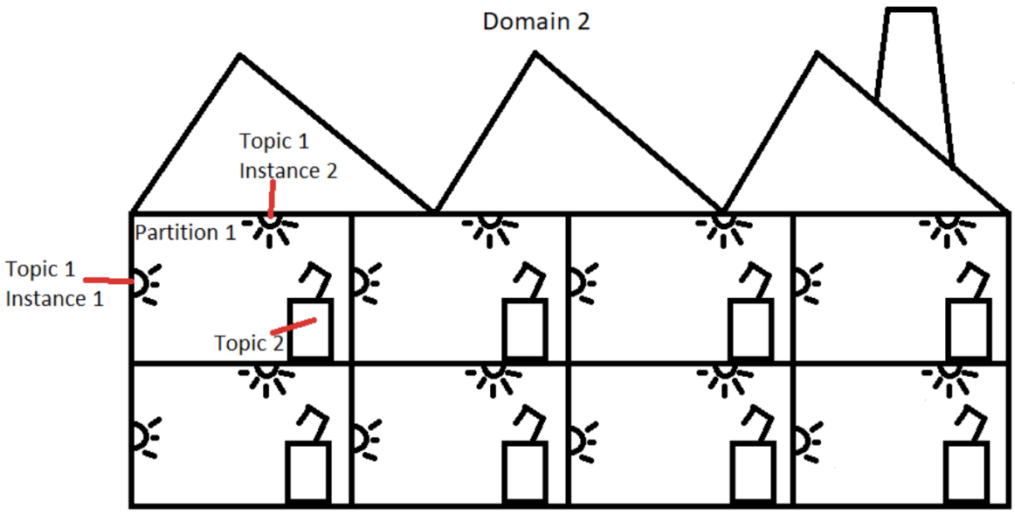

Figure 3: Factory with DDS.

Domain:
A domain is a virtual network (DDS Databus/Global Data Space) that links all DDS-applications that joined the domain. It cannot communicate with other domains unless explicitly set by the application. When you look at the example, each factory would be a domain.

Partition:
Partitions are groups within a domain. Partitions are a way of scoping data, hereby you can partition the data that belongs to each other.
Imagine that you have a building where each room has multiple sensors. You could make a partition for each room; within this partition you can make different topics that sensors write the data to.

Topic:
A Topic is a group within a partition. A topic defines the name and data type under which data is published or read. These topics can be ‘keyed’ to store different types of data. A topic receives only the dataflow they are interested in, and not data from other publications.

For example, you can make different topics for the sensors and the cobot.

Instance:

Instances are different topics with the same data types, but with different topic keys. Hereby, you can use a topic for different systems. Imagine a room with four of the same sensors. Four topics can be made, each for every sensor. However, it is more convenient to create one topic with four instances, one for each sensor. The advantage of using instances instead of creating a new topic, is that the corresponding entity is already created and discovered. Consequently, there is less memory usage, and no new discovery. (eProsima, 2019)

Sample:

A sample is a snapshot of the data of a topic or instance.

Quality of Service:

Quality of Service is a set of configurable settings that controls the behavior of the DDS system.

Participant:

To join the domain, the DDS-application must create a participant. Once the participant joins the domain, a declaration message is sent. This message contains its unique identification, key, IP-address, and its QoS settings.

Publisher:

A publisher manages the activities of the different Data Writers. The publisher determines if the data must be merged before it is sent.

Data writer:

A Data Writer is the part that prepares the data to be sent. For each topic to which data must be written, a Data Writer must be created.

Subscriber:

A subscriber manages the activities of several Data Readers. In doing so, the subscriber determines when the data received should be available to the application through the Data Readers.

Data Reader:

A Data Reader is the component that subscribes to a topic to then receive updated data from it.

1.3. DDS Architecture

When you look at the OSI model, DDS qualifies in the upper four layers.

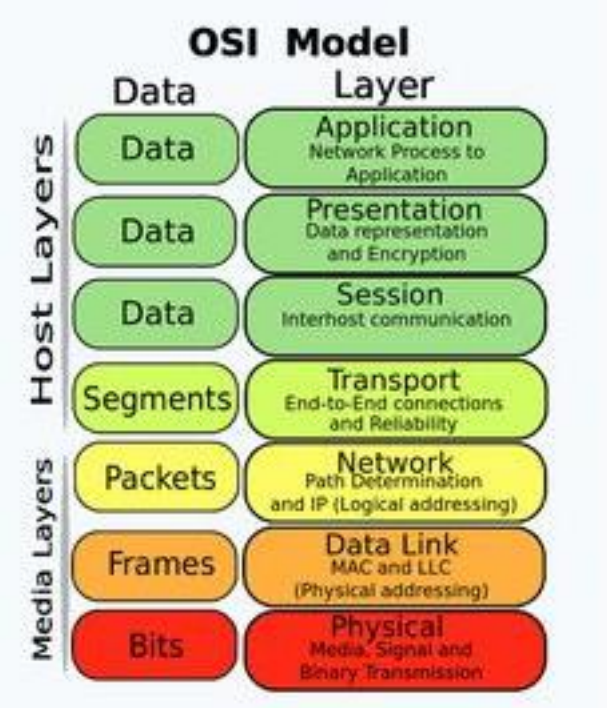

Figure 4: OSI Model.

- Transport Layer: DDS implements an RTPS protocol for sending data packets. This is based on UDP, so it uses the transport layer.
- Session Layer: DDS handles sessions within the network through domains and partitions where participants can join but also leave the session.
- Presentation Layer: DDS serializes and deserializes data using Common Data Representation (CDR). The CDR specifies the ways in which a data sample of a given type is communicated over the network.
- Application Layer: Depending on which variant of DDS being used, there is an API that allows the appropriate settings for DDS to be chosen by the user.

DDS Protocol Stack:
As mentioned before, DDS is a machine-to-machine communication protocol. DDS enabled data-exchange via publish-subscribe methodology.

As shown in figure 5, there are two layers: DCPS and DLRL. DDS V1.2 API standard is language independent, OS and HW architecture independent. The figure-5 depicts DDS protocol stack. As shown, there are two layers visualized, DCPS and DLRL.

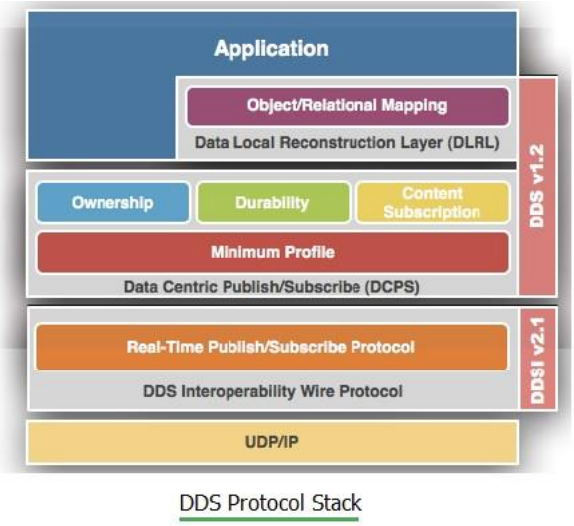

Figure 5: DDS Protocol Stack.

- **DCPS (Data Centric Publish Subscribe)** layer delivers information to subscribers. DCPS is a standard API for data centric, topic based, real time publish/subscribe layer.
- **DLRL (Data Local Reconstruction Layer)** provides interface to DCPS functionalities. This enables sharing of distributed data among devices which are IoT enabled. DLRL is standard API for creating object views out of collection of topics.

DDSI/RTPS V2.1 is a standard wire protocol which allows interoperability between different implementations of DDS standard.

DDS Protocol Architecture:

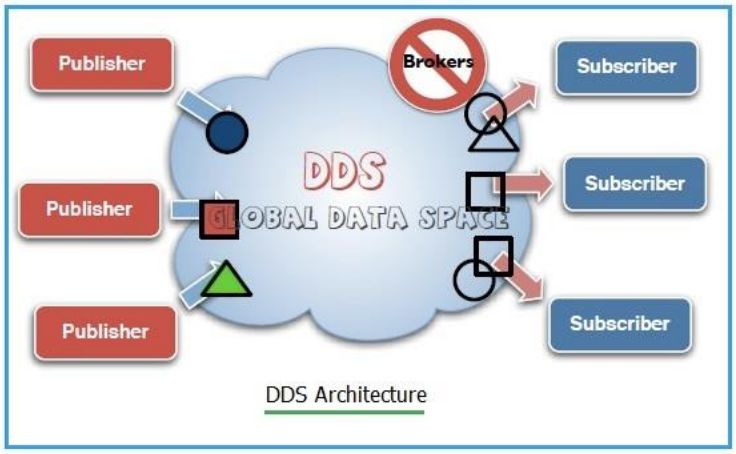

Figure 6: DDS Protocol Architecture.

DDS is a fully distributed GDS (Global Data Space). GDS specifications make it fully distributed to avoid introduction of single point of failure or bottleneck. In DDS protocol architecture, applications can autonomously and asynchronously read/write data in GDS. The publishers and subscribers can join or leave the GDS at any point in time. This is because they are dynamically discovered. Publishers and subscribers express their intention to produce or consume specific type of data such as topics. (RF-Wireless World)

1.4. DDS Databus Architecture

In intelligent distributed systems, managing dataflow is critically important. In fact, it is so important that there is an architecture dedicated to do this specific task. This architecture is a databus. (RTI, sd)

A databus is a data-centric software framework for distributing and managing real-time data in intelligent distributed systems. It allows applications and devices to work together as one, integrated system. Engineers can create multiple (even hundreds) of DDS-based layers (databuses) to separate, isolate, and selectively share communications.

Difference between a database and databus
Figure 7 describes the differences between a database and databus. The database and databus both play a role in optimizing data management in modern industrial systems.

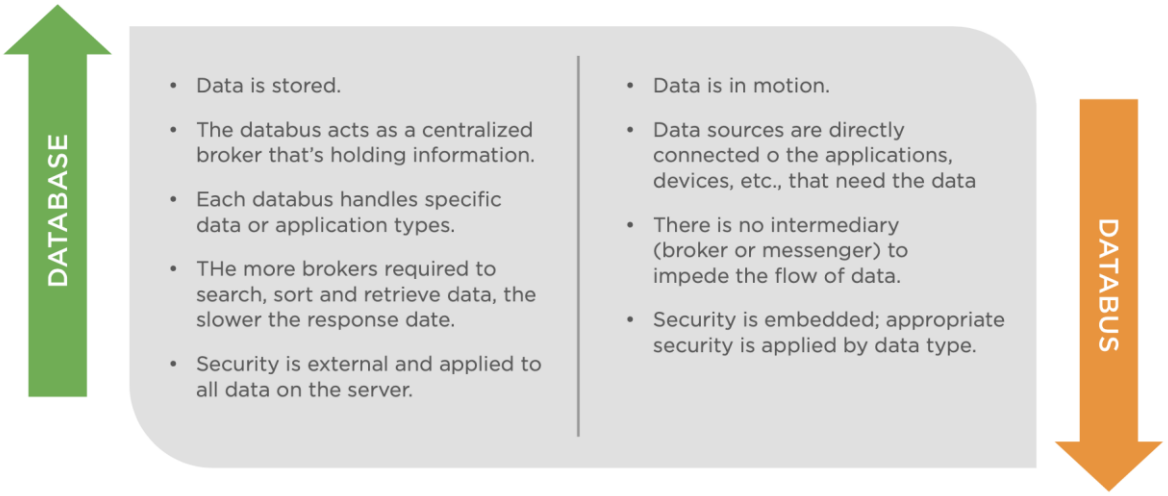

Figure 7: Differences between a database and databus.

The DDS Databus:

DDS is a communication protocol; the DDS communication layer is based on a software databus that runs via a publish/subscribe model. The DDS databus is essentially a shared global space where data is continuously flowing to and from its intended and authorized recipient(s), multiplied by the hundreds, thousands or millions of publishers or subscribers. Following is just some of the notable characteristics of a databus:

- Data-centricity: This refers to data awareness that enables things like intelligent filtering of data and delivering data to the right location at the right time.
- Built-in-QoS: QoS parameters can include such information as the rate at which the samples or data will be republished, the reliability of the delivery and the security of data flow.
- Shared data model: The shared data model eliminates data brokers or intermediaries and leads to scalability and expandability in multi-layered databases.
- Automatic discovery: A new application or device joining the databus can be provided with previously published state data.
- Integrated system: All applications, devices, modules, and subsystems work together as one integrated system.
- Security: The DDS-standard includes a plug-in security architecture that connects to a DDS library. This library includes numerous mechanisms:
  - Discovery authentication
  - Data-centric access control
  - Cryptography
  - Data tagging and logging
  - Non-repudiation
  - Secure multicasting

When there is more than one DDS databus in a design - and there is a need to share information across those databuses - the **layered databus** comes into play. This is a multi-domain databus architecture where different domains are served by their own databuses, with the ability to interconnect and enable communications between layers. The information

that is shared between the databuses is based on the DDS publish/subscribe method. A layered databus can simplify potential complexity. (RTI, 2020)

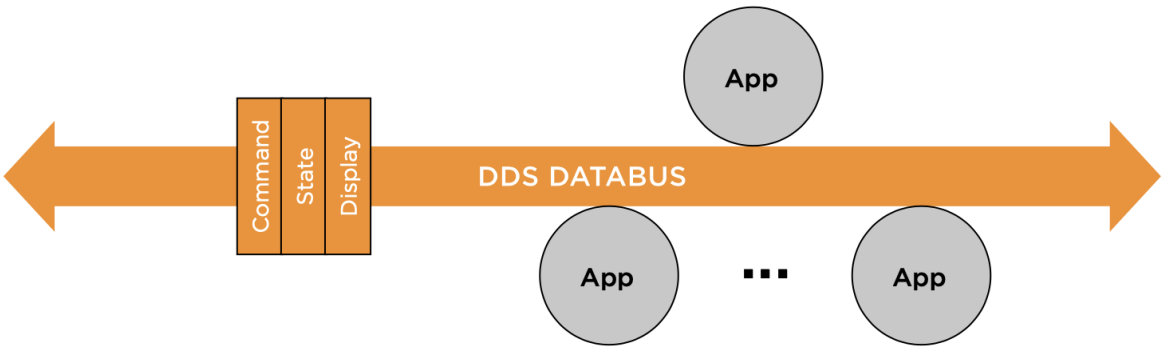

Figure 8: A single DDS databus layer.

A single DDS databus layer might be defined by its function, such as command and control. Layers can also be defined by a physical location, communication protocol, behavior, or other characteristics.

For more information about the building blocks of a layered databus you can read this whitepaper.

1.5. Importance of DDS and its security

DDS makes it easier for complex networks to share their data and expand their networks. This is possible because DDS is data centric. The goal is to make data easily accessible for all the devices on a network.

DDS provides QoS-controlled data-sharing. Applications communicate by publishing and subscribing to Topics identified by their Topic name. Subscriptions can specify time and content filters and get only a subset of the data being published on the Topic. Different DDS Domains are completely independent of each other. DDS is uniquely data-centric, which is ideal for the Industrial Internet of Things. Most middleware works by sending information between applications and systems. Data centricity ensures that all messages include the contextual information an application needs to understand the data it receives. The essence of data centricity is that DDS knows what data it stores and controls how to share that data.

The security of the DDS protocol is also very important. Let's continue with the example in chapter 1.1 about the self-driving car. Imagine that the car is programmed to turn right when you steer to the right. If the DDS protocol within your car is hacked, it can let you turn in a completely different direction. In other words, it can make your car crash, that is why the security of this protocol should be mandatory.

1.6. DDS implementations

There are a lot of different DDS implementations, the following table lists the DDS implementations we analyzed in this research. (Trend Micro Research, 2022)

Product name	Developer	Open source	Core language	Developed since
Fast-DDS	eProsima	Apache Licence 2.0	C++	2014
Cyclone DDS	Eclipse Foundation project, driven by ADLINK	Eclipse Public License 2.0 and Eclipse Development License 1.0	C	2011
OpenDDS	OCI	Custom	C++	2005
Connext DDS	RTI	Extensions are open source	C++	2005 (NDDS – 1995)
CoreDX DDS	TwinOaks	Not open source	C	2009
Gurum DDS	GurumNetworks	Not open source	C	

Table 1: DDS implementations.

2. DDS use cases

The development of DDS has started back in 2001 and has been published officially in 2004 by the OMG (Object Management Group). Since this time, DDS has been widely implemented in a lot of industries. This is because DDS is a simple, yet powerful protocol that can be used in different type of industries. To get an understanding of why and how DDS is used, there will be some use cases below that explain the use of this protocol in different organizations. In the last part, an example will be given about how DDS is used in practice.

2.1. Defense industry

NASA
The first use case that will be mentioned is NASA. NASA has implemented DDS in different ways in their infrastructure. “The most notable use of DDS in this sector is NASA’s launch control system at Kennedy Space Center (KSC) using one of the world’s largest supervisory control and data acquisition (SCADA) systems with over 400,000 control points. Other use cases are for C&C services (such as for bridging Ethernet networks to tactical radio equipment on the field), which requires efficient and reliable data transport in challenging conditions. “ (Trend Micro Research, 2022)

Army
“Some system integrators such as MilSoft and KONGSBERG specialize in using DDS for defense applications. The Spanish Army uses Fast-DDS for C&C applications, while an unidentified defense technology company uses OpenDDS for its connectivity framework. “ (Trend Micro Research, 2022)

2.2. Energy

“DDS is also used by Siemens and Sunrise Wind for example in the energy sector for power generation and distribution. DDS is being used in this field and is predicted to become more popular in the future. For instance, OpenSplice Vortex (now ADLINK Cyclone DDS) is used at the large-scale fusion reactor system in the Culham Centre for Fusion Energy, RTI Connext DDS is used in distributed power generation by Siemens, and LocalGrid uses DDS for smart-grid distribution and control and for research.” (Trend Micro Research, 2022)

DDS is used in this sector because the energy sector is undergoing significant changes and innovation. Owing to emerging and available IIoT solutions, an entire spectrum of issues is covered, such as clean and traditional power generation, storage and management, and distribution from companies themselves to the end users being adopted by states.

2.3. Health care

“DDS enables interoperable data connectivity for medical devices and clinical systems. For example, it is used by the MD PnP interoperability program, which facilitates the adoption of open standards and interoperable technologies in integrated clinical environments (ICE). Companies such as GE Healthcare use this connectivity to work with over 200 hospitals with a command center software in different countries with its applications to process real-time

needs. RTI Connext DDS is used in MRI and CT scanning equipment and for hospital patient safety and integration of clinical decision systems. ADLINK's OpenSplice DDS is used to integrate medical tablets and in medical panel computers." (Trend Micro Research, 2022)

2.4. IoT

"In 2020, the International Federation of Robotics (IFR) estimated that 373,000 industrial robots were installed globally, which was a huge jump compared to 2011, when they estimated 89,000. The market for professional service robots (such as for transportation, logistics, cleaning, and hospitality) grew in 2020 by 12%, with 1,067 companies specializing in service robots worldwide (a 17% increase from 2019).
DDS plays a fundamental role in the robotics sector because it is the default middleware of ROS 2, which is the rapidly growing, de facto standard OS for consumer, service, and industrial robots, as well as for autonomous systems in general. Some would say that ROS is for robotics what Ubuntu and Linux is for computing. Particularly, Eclipse Cyclone DDS has been chosen to be the default DDS implementation in ROS 2.

From 2019, the adoption rate of ROS 2 went from less than 5% to more than 50%. According to ROS Metrics, 55% of the downloads of ROS are ROS 2, which points to a DDS layer. The ROS official docker image has been downloaded more than 10 million times. AWS RoboMaker, a simulation service used by robotics developers such as iRobot, is based on ROS 2. While these numbers do not directly indicate the number of robots running ROS 2, it implies a growing trend and interest in the sector.

As more sectors make use of robots running ROS 2 for operations, more devices and machines can become vulnerable to attacks that abuse gaps in DDS. This is because if the DDS implementations are not patched periodically, they will get more vulnerable over time. As noted by the Rochester Institute of Technology, differently than with ROS, "network vulnerabilities are directly tied to the functionality in ROS2 with their new DDS standard," and their conclusion is that most of the security issues found during the research are brought in by DDS, also confirmed by other researchers." (Trend Micro Research, 2022)

2.5. Examples in practice

Some interesting best practices about the implementation of DDS can be found here. This is the official OMG website and they have posted some interesting use cases about DDS implementation in different fields.

3. DDS vulnerabilities

As mentioned in the first chapter, there are multiple DDS implementations available to use. Each of these implementations have been founded by different organizations and therefore every implementation has its own vulnerabilities and mitigations. It's important to note that DDS has known vulnerabilities that have been patched already and there are new DDS vulnerabilities that haven't been patched yet.

Trend Micro has noted a couple of attacks that have been executed on the DDS protocol by other researchers, but they have also done some vulnerability scanning themselves. In the paragraph below, the vulnerabilities have been divided into two parts, the first will be about the vulnerabilities that have been found by other researchers and the second will be the findings from Trend Micro.

3.1. Known DDS vulnerabilities

RTI Connext DDS
There are several vulnerabilities that have been found in RTI Connext DDS. All these vulnerabilities that have been found have to do with T0866: Exploitation of Remote Services. The weaknesses that have been found are CWE-122/190/120 and can result in a buffer overflow. Another one is T0816: DoS, the weakness that has been found here is possible via CWE-502: Deserialization of untrusted data. The reference number for these vulnerabilities is 2017-A-0097.NASL.

Cyclone DDS
For cyclone DDS, there are two unpatched vulnerabilities. The first one is T0814: Denial of Service and is connected to the weakness CWE-787: Out-of-bounds write. The references for these are CVE-2020-18734/18735.

3.2. New DDS vulnerabilities

Fast DDS
EProsima Fast DDS versions prior to 2.4.0, are susceptible to exploitation when an attacker sends a specially crafted packet to flood a target device with unwanted traffic, which may result in a DoS condition and information exposure. This vulnerability has to do with T0804: Brute Force and the reference for this is CVE-2021-38425.

OpenDDS
Some other vulnerabilities that have been found by Trend Micro have to do with malformed RTPS packets. The researchers were able to trigger a DoS and remotely execute arbitrary code. This vulnerability has to do with T0802: Automated collection and T0846: Remote System Discovery. The references for these vulnerabilities are CVE-2021-38445 and CVE-2021-38447.

RTI Connext DDS
RTI Connext DDS Professional, Connext DDS Secure versions 4.2x to 6.1.0, and Connext DDS Micro versions 2.4 and later are vulnerable when an attacker sends a specially crafted packet

to flood target devices with unwanted traffic. This may result in a DoS condition and information exposure. This vulnerability has to do with T0827: Loss of Control and the reference for this is CVE-2021-38487. Another vulnerability has to do with T0862: Supply Chain Compromise and through this vulnerability it's possible to do a buffer overflow. The reference for this is CVE-2021-38427 and CVE-2021-38433.

GurumDDS

All versions of GurumDDS are vulnerable to heap-based buffer overflow, which may cause a DoS condition or remotely execute arbitrary code. The reference for this vulnerability is CVE-2021-38439 and has to do with T0856: Spoof of Reporting Message.

CycloneDDS

In CycloneDDS, a few vulnerabilities have been found that can cause to execute arbitrary values in the XML parser. The references for these vulnerabilities are CVE-2021-38443 and CVE-2021-38441.

References

DDS-Foundation. (n.d.). *How DDS works.* Retrieved from DDS-Foundation: https://www.dds-foundation.org/how-dds-works/

DDS-foundation. (n.d.). *How does DDS work?* Retrieved from DDS Foundation: https://www.dds-foundation.org/how-dds-works/

DDS-Foundation. (n.d.). *What is dds.* Retrieved from DDS-foundation: https://www.dds-foundation.org/what-is-dds-3/

eProsima. (2019). *3.5.1. Topics, keys and instances.* Retrieved from Fast DDS: https://fast-rtps.docs.eprosima.com/en/latest/fastdds/dds_layer/topic/instances.html

Helmer, T. (2021). *Onderzoeksverslag werkcel met DDS.*

RF-Wireless World. (n.d.). *DDS Protocol Architecture basics | DDS Protocol in IoT.* Retrieved from RF-Wireless World: https://www.rfwireless-world.com/Terminology/DDS-protocol-architecture.html

RTI. (2020). *The Layered Databus Architecture .* Retrieved from RTI: https://www.dds-foundation.org/wp-content/uploads/2020/10/The-Layered-Databus-Architecture-white-paper-2020.pdf

RTI. (n.d.). *What is a databus?* Retrieved from RTI: https://www.rti.com/products/what-is-a-databus

Trend Micro Research. (2022, April 19). *A Security Analysis of the Data Distribution Service (DDS) Protocol.* Retrieved from Trend Micro Research: https://documents.trendmicro.com/assets/white_papers/wp-a-security-analysis-of-the-data-distribution-service-dds-protocol.pdf

Wikipedia. (n.d.). *Data Distribution Service.* Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Data_Distribution_Service#cite_note-12